Formalization of a cooperation model based on joint intentions

Brazier, F.M.; Jonker, C.M.; Treur, J.

published in
Intelligent Agents III, Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96,
1997

document version
Publisher's PDF, also known as Version of record

Formalization of a cooperation model based on joint intentions*

F.M.T. Brazier, C.M. Jonker, J. Treur

Vrije Universiteit Amsterdam
Department of Mathematics and Computer Science
Artificial Intelligence Group
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Fax: +31.20.4447653 Email: {frances,jonker,treur}@cs.vu.nl

1 Introduction

Distributed project coordination requires insight in the types of interaction involved in engineering practice. In current practice, well-structured hierarchical management and decentralised project organisation are often combined. Within an organisation, a number of levels can be found within which responsibility for effective interaction is delegated to the engineers themselves. Engineers decide when to exchange preliminary ideas and partial designs, when to acknowledge possible conflicts and when to resolve such conflicts, when to question requirements, et cetera. A combination of traditional management structures and virtual organisations result in dynamic structures, liable to considerable change during the life span of a project.

The types of interaction encountered in such real-life engineering situations show how intricate such processes can be. Within the multi-agent community the problem of distributed problem solving has been recognised; see for example (Dunskus, Grecu, Brown and Berker, 1995; Petrie, 1994). In (Jennings, 1995) an informal multi-agent model for cooperative problem solving is proposed and its implementation in one specific environment is described. This model was developed in the context of the ARCHON project which focussed on electricity transportation management (Jennings, Corera, Laresgoiti, Mamdani, Perriolat, Skarek, Varga, 1995). Essential elements of this model are the dynamic organisation and management of joint activities, susceptive to change due to unexpected events. As described, the model, however, does not provide enough detail to support analysis, modelling, reuse, and implementation of coordination systems in specific domains. In this paper a formal model is proposed which does provide the level of detail required. This model, a formalization of Jennings' model, is more refined and more generic than Jenning's model. It is more refined in the sense that more detail of the organisation and management of joint projects is included. It is more generic in the sense that domain specific knowledge and domain independent aspects have been clearly separated, which simplifies reuse. The model is described in the DESIRE framework (Langevelde, Philipsen and Treur, 1992; Brazier, Dunin-Keplicz, Jennings and Treur, 1995; Brazier, Treur, Wijngaards and Willems, 1995, 1996), a framework for the design and (formal) specification of complex compositional systems. DESIRE is briefly introduced in Section 2. In Section 3 a specification of a generic agent model is introduced. In Section 4 this model is specialised to a more refined agent model for a cooperative agent based on Jennings' model of cooperation. In Section 5 a brief synopsis is given of an application of the refined agent model for a real design project in which traditional management and virtual organisations are combined: the design of part of the interior of a specific aircraft.

2 Specification of Multi-Agent Systems

In projects such as the design project sketched above, task coordination between agents is essential. As agents, however, often perform more than one task, (sequentially or in parallel), task coordination

Brazier, F.M.T., Jonker, C.M., Treur, J. (1997). Formalisation of a cooperation model based on joint intentions. In: J.P. Müller, M.J. Wooldridge, N.R. Jennings (eds.), Intelligent Agents III (Proc. of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96), Lecture Notes in AI, volume 1193, Springer Verlag, pp. 141-155.

within the agents themselves is also of importance. Within the formal compositional framework DESIRE (Langevelde, Philipsen and Treur, 1992; Brazier, Treur, Wijngaards and Willems, 1995; Brazier, Dunin-Keplicz, Jennings and Treur, 1995, 1996) task models are used to define compositional architectures. Task models include knowledge of

(1) a task (de)composition,
(2) information exchange,
(3) sequencing of (sub)tasks,
(4) sub-task delegation, and
(5) knowledge structures,

These five types of knowledge are explicitly modelled and specified at different levels of abstraction. Tasks are defined at different levels of abstraction, resulting in a task (de)composition. Different levels of abstraction are distinguished within knowledge structures; for example taxonomies of information types. Tasks refer to these knowledge structures. Sequencing of tasks and goals, and information exchange reflect the abstraction level of tasks involved. Task delegation, the last of the five types of knowledge, is also defined at all levels of abstraction within a task model. More abstract tasks may be delegated to more than one party, whereas more specific tasks are often delegated to one particular party.

The model of cooperation presented in this paper has been formally specified within the DESIRE framework. The semantics of the formal specification language are well-defined, based on temporal logic; see (Brazier, Treur, Wijngaards and Willems, 1996). By explicitly modelling and specifying the semantics of static and dynamic aspects of a system, a well-defined conceptual description is acquired that can be used for verification and validation, but also is a basis for reuse. Translation to an operational system is straightforward; the framework, in fact, includes implementation generators with which formal specifications can be translated into executable code. DESIRE has been successfully applied to design and develop both single agent and multi-agent systems (Brazier, Dunin-Keplicz, Jennings and Treur, 1995).

3 A Generic Model of an Agent

A cooperative agent performs a number of generic tasks. Some of these tasks deal with the relationship of an agent to the world: maintaining information about the world (*world model*), and managing interaction with the world (*observation*, *execution of actions* that change the world). Other tasks concern its relationship to other agents: maintaining information on other agents (*agent models*), managing interaction with other agents (*communication*), and managing activities performed jointly with other agents (*cooperation*). Furthermore, tasks of a more reflective nature are performed: maintaining information of an agent's own processes over time (*history*), and managing an agent's own processes (*own process control*). In addition to these generic tasks, agent specific tasks are distinguished: tasks that may differ between agents (*agent specific tasks*).

Each of the eight generic agent tasks distinguished is specified by a *component* at the top level of the agent: agent_specific_tasks (AST), own_process_control (OPC), maintain_history (MH), agent_interaction_management (AIM), maintain_agent_information (MAI), cooperation_management (CM), world_interaction_management (WIM), and maintain_world_information (MWI); a graphical representation is shown in Figure 1.

The agent component OPC is responsible for determining, planning, scheduling and monitoring an agent's activities. Furthermore, it is responsible for maintaining all relevant information on the agent's activities and its status. AST is mostly domain-specific and may differ per agent. It contains a task-hierarchy and knowledge necessary to perform tasks in interaction with other components of the same agent. The component MH is responsible for the storage of the sequences of internal and external processes of an agent, for which purposes and with which results. Upon request, part of this information can be sent to other components or agents. Information of this kind is useful in strategic reasoning. For example, if a goal can be reached via different recipes and one of these recipes has previously been attempted and failed, another recipe should be attempted.

The component AIM manages communication with other agents, in particular with team members of a project. It receives information from CM which it transfers to (possible) participants in a project. Furthermore, it receives (communicated) information from other agents which it transfers to other

relevant components. In MAI information on other agents is stored that, if required, can be made available to other components. The component CM is responsible for all tasks concerning projects, project commitments and cooperation. The component MWI contains the current world state as known to the agent. It stores all information obtained by monitoring the world. The component WIM is responsible for the execution of observations and actions. An important sub-task of this component is the observation of the effects on the world of the tasks executed by the other agents and by the agent itself.

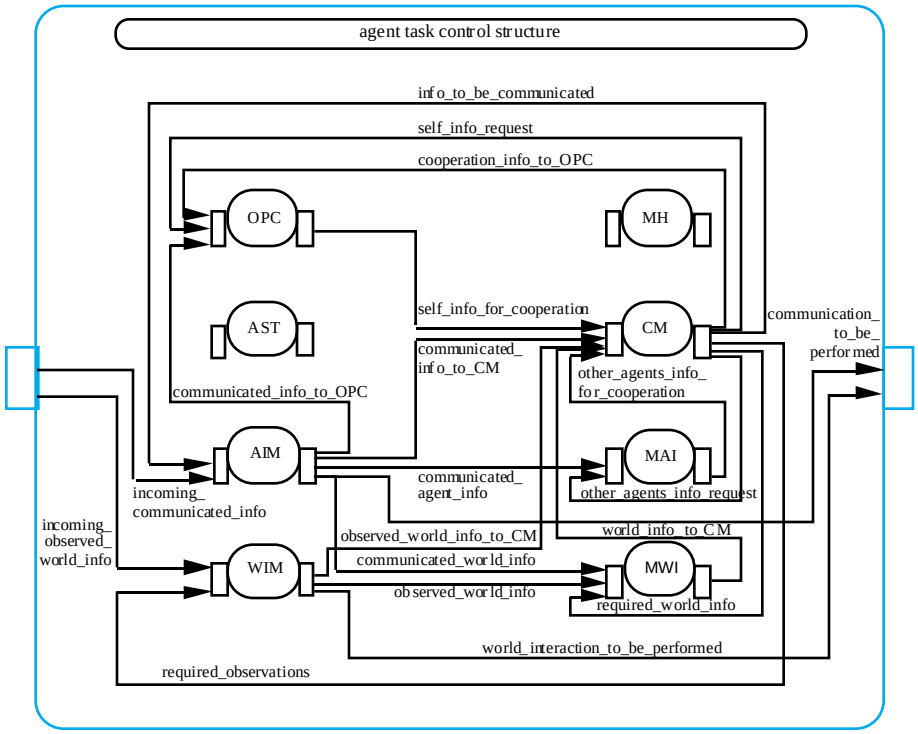

Figure 1 Component and interaction structure at the top level of a cooperative agent

Information links are defined to specify *information exchange* between agents, between agents and the world, and between components within an agent. In Figure 1 the information links at the top level of the agent are depicted as arrows. For clarity only those links are considered that are relevant for the cooperation model. In general, information communicated to an agent by another agent, is transferred directly from the input interface to the agent's component AIM by the link incoming_communicated_info. Comparably information to be communicated from an agent to other agents is determined by the agent's component AIM and is transferred from this component to the output interface of the agent by the link communication_to_be_performed.

A similar role is played by the component WIM in the interaction of the agent with the world: observations and actions to be executed in the world are determined by the agent's component WIM and transferred from this component to the agent's output interface by the link world_interaction_to_be_performed. Observation information received by the agent from the world is transferred from the agent's input interface to the component WIM by the link incoming_observed_world_info.

From the component WIM the incoming information can be transferred further to other components via the links observed_world_info (to the component MWI , where the agent stores its information of the current world state) and observed_world_info_to_CM. The incoming communicated information can be transferred from the component AIM to other components by the links communicated_info_to_OPC, communicated_info_to_CM, communicated_world_info, and communicated_agent_info. The component CM requires information on the agent itself and on other agents. This information is provided, respectively, via the links self_info_for_cooperation and other_agents_info_for_cooperation. Supply of information of this kind only takes place when required. To this end the links self_info_request and

other_agents_info_request transfer meta-information on which specific information is required. The explicit formulation in DESIRE of which information is to be transferred and from where to where makes fine-grained control of the information exchange within the agent possible.

The information on the cooperation as determined by the component CM has to be provided to the agent itself and to the other agents involved in the cooperation. The former is performed by the link cooperation_info_to_OPC. The latter involves the component AIM. The link info_to_be_communicated transfers the information to this component. In AIM communication to be performed is prepared. This information is transferred to the agent's output interface by the link communication_to_be_performed (from where the information is transferred to the agents to whom the information is addressed). The component CM also requires information on the world. This information can be supplied in a controlled manner by the agent's storage of world information in the component MWI, using the links required_world_info and world_info_to_CM. However, if the required world information is not yet available within the agent, OPC may determine that an action is required to acquire the information from outside. This can be performed by a controlled observation of the world by the agent. To this end the links required_observations and observed_world_info_to_CM are specified. Of course, it is also possible to obtain information on the world by communication with other agents. This is modelled by the links info_to_be_communicated (transferring a request for world information for another agent) and communicated_info_to_CM (transferring world information received from another agent) discussed above.

Task control is specified by task control knowledge within agents and within components of agents. This control knowledge explicitly expresses which components should be activated when and how, which goals are associated with component activation, and the amount of effort which can be afforded to achieve a goal to a given extent. These aspects are specified as component and link activation together with sets of targets and requests, exhaustiveness and effort to define the component's goals. Once an agent has been awakened, some components of the agent (for instance, OPC and CM) are permanently activated by task control rules of the form

if start
then next_component_state(*component_name*,**awake**)

Components that are not always awake, are activated with rules of the form

if *activation_condition*
then next_component_state(*component_name*,**active**)
and next_target_set(*target_set_name*)

The component is only active while trying to derive the information indicated by *target_set_name*. The activation condition specifies which target_sets must be fulfilled, by which components and with which exhaustiveness. Examples are to be found in Section 4.4.

4 Specification of the Cooperation Model

In this section, the generic agent model of Section 3 is refined to accomodate the cooperation model. For detailed specifications see (Brazier, Jonker and Treur, 1996).

In Jennings' model of cooperation, agents are capable of organising projects. An agent decides to organise a project to reach a given goal. With respect to the current state of the world, an agent determines a set of activities to reach this goal and the temporal dependencies between the activities. In interaction with these agents, the organising agent determines which agents are willing and able to participate in the project. On the basis of this information, the activities to be performed, the order in which the activities are to be performed and the deadline, the organising agent tries to put together a project team and a project schedule (called a *recipe*). The creation of this recipe is an iterative process requiring interaction with the other agents on their own schedules (related to other projects). When completed, the recipe is sent to all participants, and the project commences.

Once committed, each participating agent (including the organiser) receives the final recipe, and is committed to the relevant time interval in the recipe. Each agent has the same obligation towards the project: each monitors the progress of the project and is equally responsible for its success. If a team-member discovers a problem that endangers the project, he/she informs all participants. One of the agents (e.g., the project manager) can then take the initiative to modify the project plan, to create a new project for the same goal or to inform all participants that the goal is unattainable or that it is no longer necessary to reach the goal.

In Figure 2 a hierarchical task decomposition for a cooperative agent equipped with the model of cooperation is depicted as a specialisation of the agent model introduced in Section 3.

In Sections 4.1 to 4.5 the agent components that play an important role in the cooperation are described in more detail.

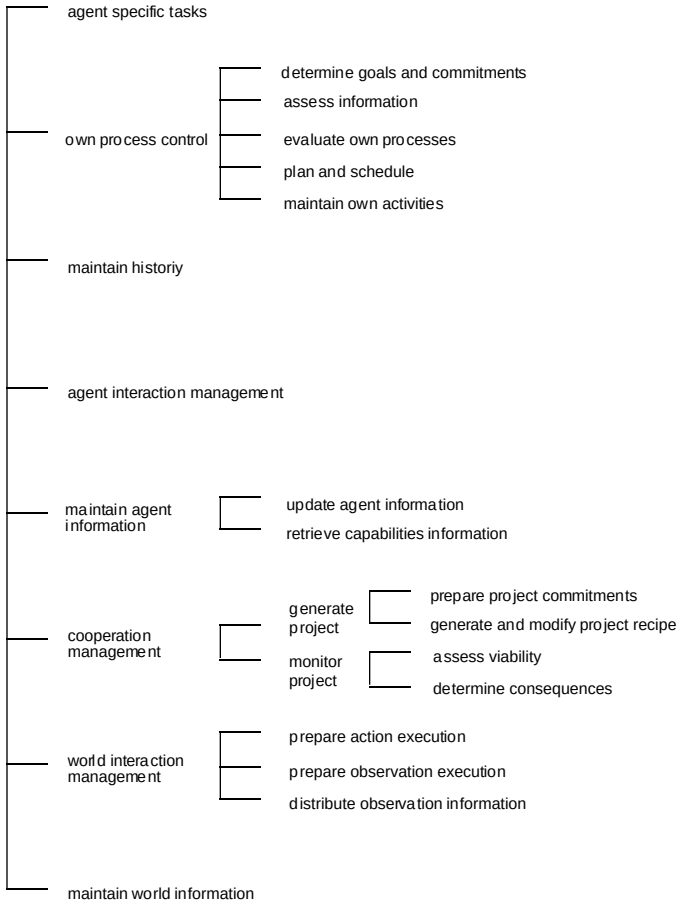

Figure 2 Task hierarchy for a cooperative agent

4.1 Own Process Control (OPC)

The agent component OPC is a composed component responsible for determining, planning, scheduling and monitoring an agent's activities. Furthermore, it is responsible for maintaining all relevant information on the agent's activities and its status. These sub-tasks are performed by OPC's sub-components: determine_goals_and_commitments (DPC), assess_ information (AI), evaluate_own_processes (EOP), plan_and_schedule (PS) and maintain_own_activities (MOA).

4.1.1 Determine Goals and Commitments (DGC)

Component DGC determines goals of an agent on the basis of its motivations, priorities, deadlines, and its role within a system. Selection of a goal depends on motivation: motivation is a necessary precondition for goal selection. Selection of a goal implies individual commitment to the goal.

4.1.2 Assess Information (AI)

The AI component maintains all relevant information on an agent's activities: which information is based on its own observations; which on own assumptions; which has been received by communication, and from which source; and which information has been derived, and is based on which other information.

4.1.3 Evaluate Own Processes (EOP)

Component EOP is responsible for the evaluation of the progress of an agent's activities with respect to its individual commitments, by monitoring relevant activities (its own and other agents) and analysing monitoring this information. During analysis EOP may, for example, deduce that the motivation for a goal has disappeared: this goal is then removed.

4.1.4 Plan and Schedule (PS)

The component PS is responsible for planning and scheduling an agent's activities, upon request for participation in a project by another agent or on the basis of information received from EOP or DGC. The component PS uses domain-knowledge to find a set A of activities, called a plan, that meets the following criteria: (1) execution of the plan will lead to the fullfilment of a goal G, (2) the plan can be scheduled without contradicting prior commitments, (3) the plan matches the priority and the deadline of the goal. If no such plan and schedule can be found, not even by requesting the help of other agents, this must be communicated to EOP. Another goal can then be selected by DGC. If an agent cannot reach the goal G itself while respecting the priority and deadline, but the goal may possibly be reached with the help of others, then all relevant information is sent to CM, which will try to create a project to reach the goal.

4.1.5 Maintain Own Activities (MOA)

This component stores an agent's own schedule, which actions an agent can perform (domain dependent) and which commitments an agent has made to which goals. Commitments can be made with respect to other agents and projects.

4.2 Agent Interaction Management (AIM)

As discussed in Section 3, the component AIM manages communication with other agents, in particular with team members of a project. For example, upon receiving a new recipe, AIM determines the subset of recipe-elements that concern its own activities. This subset is passed on as "own process" information to OPC. The whole recipe is sent to CM.

4.3 Maintain Agent Information (MAI)

Upon request MAI provides other agents or other sub-components with names of agents capable of performing certain specified activities. Two sub-components are responsible for the performance of this task: update_agent_information (UAI) and retrieve_capabilities_information (RCI).

4.3.1 Update Agent Information (UAI)

UAI maintains models of other agents known to an agent itself. A model of another agent consists of statements that express cooperativeness of the other agent, its availability (that it normally has no time to help other agents, or normally is able to help), punctuality with respect to deadlines, et cetera. UAI stores and updates its knowledge by maintaining which activities other agents are capable of performing, the projects in which they participate and the goals to which they are committed.

4.3.2 Retrieve Capabilities Information (RCI)

RCI provides, for each activity, the names of all agents known to be capable of performing an activity and the available meta-information concerning the exhaustiveness of the information.

4.4 Cooperation Management (CM)

The component CM is a composed component responsible for all tasks concerning projects, project commitments and cooperation. Before describing CM's subcomponents in detail, the information links within CM and part of CM's control structure are given.

The interaction between the components of CM and CM's environment is organized through the links depicted in Figure 3. If a new project is to be created the relevant information enters the component GP through the link required_project (which transfers the information also transferred by link self_info_for_cooperation of Figure 1). The information GP needs on other agents enters GP through link info_on_other_agents (see also links other_agents_info_for_cooperation and communicated_info_to_CM of Figure 1) and this information is requested through link required_info_on_other_agents (see also links info_to_be_communicated and self_info_request of Figure 1). The commitments made in the created project and the information on the joint project are transferred through link commitments_to_output (see also links info_to_be_communicated and cooperation_info_to_OPC of Figure 1). The generated project is sent to MP to be monitored through link own_generated_project. If GP is activated because a monitored project is to be reconsidered, the relevant information is transferred through link monitoring_info. The links incoming_project_info (see also link communicated_info_to_CM of Figure 1) and own_generated_project transfer the necessary information on projects that MP has to monitor. For this purpose MP needs information which is requested through the link required_monitoring_info (see also links required _observations and required_world_info of Figure 1) and enters MP through link incoming_project_info (see also links

observed_world_info_to_CM and world_info_to_CM of Figure 1). If necessary the resulting monitoring information is transferred through the links monitoring_info and monitoring_info_to_output (see also links info_to_be_communicated and cooperation_info_to_OPC of Figure 1).

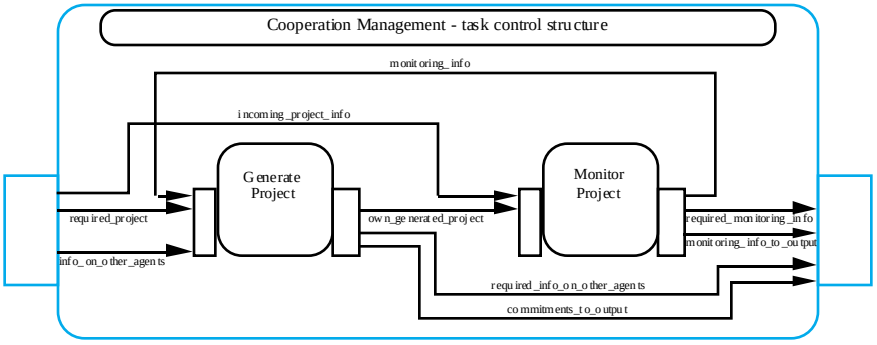

Figure 3 Components and Interaction within the cooperation management component

The control structure of CM contains the following rules.

if start
then next_component_state(monitor_project,**awake**)
and next_link_state(incoming_project_info,**awake**)

if target_set(project_to_be_generated)
and component_state(generate_project,**idle**)
then next_component_state(generate_project,**active**)
and next_target_set(project_to_be_created)
and next_link_state(required_project,**up_to_date**)

The occurrences of "**awake**" in the first rule expresses that the moment CM is activated, CM's subcomponent MP and the link incoming_project_info are to be activated ad infinitum. MP continuously monitors projects and for this MP needs all monitoring information as soon as it is available. The "**active**" of the second rule expresses that GP is only active while trying to generate a new project. The "**up_to_date**" expresses that the information transferred by the link required_project, should be available in the input interface of GP, prior to the activation of GP. The activation condition of the second rule is that CM is set the target "project_to_be_generated" and GP is not already active.

4.4.1 Generate Project (GP)

Given the goal G, motivation M, priority p, deadline T, all possible sets A of activities with which goal G can be reached, and an agent's own capabilities, the component GP has two main tasks: to prepare project commitments (PPC), and to generate and modify project recipes (GMR). Links are defined to regulate the interaction between GP's components and its environment, see Figure 4. Recipes enter PPC through the links recipe_to_be_repaired (see also link monitoring_info of Figure 3) and recipe_to_be_prepared (see also link required_project of Figure 3). To prepare the commitments PPC requests information on other agents. These requests are sent through the link needed_info_on_other_agents (see also link required_info_on_other_agents of Figure 3), the answers enter through link info_on_other_agents_to_PPC (and link info_on_other_agents of Figure 3). The information produced by PPC is transferred to GMR through link prepared_project. While making the recipe GMR interacts with other agents through the links info_for_agents (and link commitments_to_output of Figure 3) and info_on_participants (see also link info_on_other_agents of Figure 3). The final recipe is sent to the participants through link info_for_agents.

The component **Prepare Project Commitments (PPC)** determines a preferred set A of activities with which goal G can be reached. Using domain-knowledge the dependencies between the activities in A are determined using critical path methods. This (partial) ordering of the activities in A is important in the development of a recipe R for goal G. Given this dependency-graph PPC determines which agents can and are willing to perform activities to help reach goal G. The dependency-graph for A, the information (G, M, p, T), the relevant capabilities of the willing participants (including the agent's own relevant capabilities) and the corresponding names of the agents, are sent to GMR.

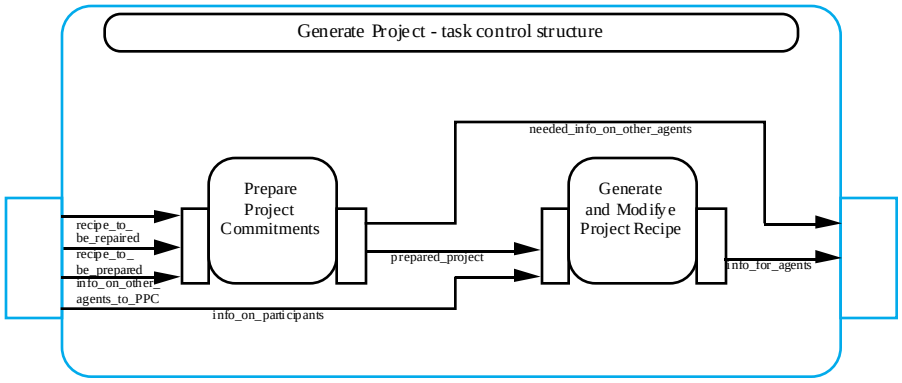

Figure 4 Components and Communication within the generate project component

Using PPC's information, the component **Generate and Modify project Recipe (GMR)** designs a recipe R that conforms to the interdependencies between the activities in A (thus leading to G's fulfilment). The recipe R is interactively designed by iteratively generating and sending proposed recipe elements to agents interested in participation. A recipe element consists of a task of A, a willing participant capable of performing that task, a priority p and a deadline T for that task. The willing participants accept, adapt or reject the proposed recipe elements. Acceptance or adaptation of a recipe element implies that the agent commits itself to this element. GMR adjusts the partial recipe depending on the replies from participating agents. A recipe may be found that is acceptable to all participants and that will reach goal G before its deadline. The duration of the recipe and team building is estimated on the basis of the number of activities involved, the number of willing participants and the time needed for communicating requests and responses. The time required for communication (depending on the situation) is assumed to be known. In addition, communication is assumed to be error free. The resulting recipe is communicated to all participants.

4.4.2 Monitor Project (MP)

The component MP is responsible for the detection of the need for alterations to a project or the need to end a project. MP monitors the progress of projects. In order to perform its task MP has two sub-components: assess_viability and determine_consequences. The components and the links for interaction within MP are depicted in Figure 5. Information on the project to be monitored and the necessary monitoring information enters AV through link project_info. The request for monitoring information and the resulting assessment information is transferred to CM's output interface through links assessment_info_to_output and monitoring_info_to_output (see Figure 3). Through link assessment_info_to_DC this information is also sent to DC, which uses it to determine changes to the joint project. Information on changes is sent through the links info_on_project_changes and monitoring_info_to_output (see Figure 3) to CM's output interface. From CM's output interface the information is transferred to AIM through the link info_to_be_communicated (see Figure 1).

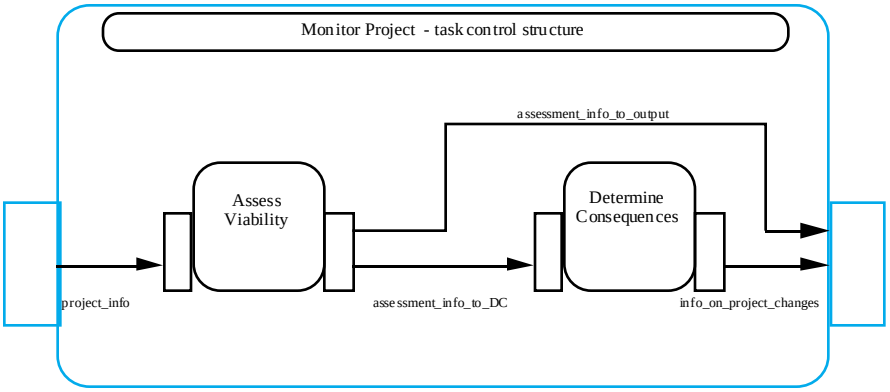

Figure 5 Components and Communication within the monitor project component

Assess Viability (AV) monitors the viability and validity of recipes. To check the validity of a project recipe, AV uses the same considerations as the sub-component EOP of the component OPC. To monitor the process it uses information received from OPC, WIM and MWI (its other components). It can also actively formulate requests for observational information from WIM, MWI or information of other agents via MAI and AIM.

Determine Consequences (DC) interprets AV's monitoring results. DC issues requests to find new recipes or to adapt existing recipes, to the component project_generation of CM and issues corresponding messages to the participants. DC also determines when a goal G should be withdrawn (for example, because the goal is unattainable, the goal has been reached, or because the motivation for the goal no longer exists) and prepares and issues a message to that effect to each participant.

4.5 World Interaction Management (WIM)

The component WIM is responsible for the execution of observations and actions. An important sub-task of this component is the observation of the effects on the world of the tasks executed by the other agents and by the agent itself.

4.5.1 Prepare Action Execution (PAE)

This component prepares the execution of actions determined by AST by communicating to the world which actions should be taken.

4.5.2 Prepare Observation Execution (POE)

WIM prepares specific observations. The observational information is sent via DOI to those sub-components that analyse this information.

4.5.3 Distribute Observation Information (DOI)

Upon request, observational information is sent from DOI to other components (including MWI). DOI can also take the initiative to inform other components (including MWI) of (domain-dependent) important changes in the world.

5 An Application of the Model to Distributed Project Coordination

In this section a simplified example of the coordination of the routine design of aircraft interior is analysed. Agents refer to individuals (or groups of individuals) with a specific task in the project. A design project manager is assigned the task of coordinating all design activities for the interior of an aircraft, for example the design of the toilet unit, luggage bins, wardrobe, gallies, side panels, and the floors, in close collaboration with the financial department. The responsibility for the design of each of the individual units is delegated to a unit manager, who in turn coordinates the design of more specific aspects of that unit. The design project manager interacts with a number of specialists: financial specialists, styling specialists, logistic specialists, tooling specialists, et cetera, to coordinate the project as a whole.

6.1 Communication between agents

Interaction between agents is modelled by information links, controlled by the agent from which the links originate. Different types of information are exchanged through links: both *object level information*, such as information on the design object description, the initial cable routing, switch dimensions and positions, the initial design, product information, and *meta-level information*; i.e., requests for information, evaluation information on the design object description, conflicts between routing of cables and the initial design, and information on the design process (e.g., planning and scheduling). The double-arrowed lines in Figure 2 depict the information links that specify the exchange of these types of information between agents.

To describe the interaction between agents the creation of a project is sketched from the perspective of a design project manager. In Table 4 a system trace is presented for the creation process, sketching the activation of agents, components of agents and the information communicated through time.

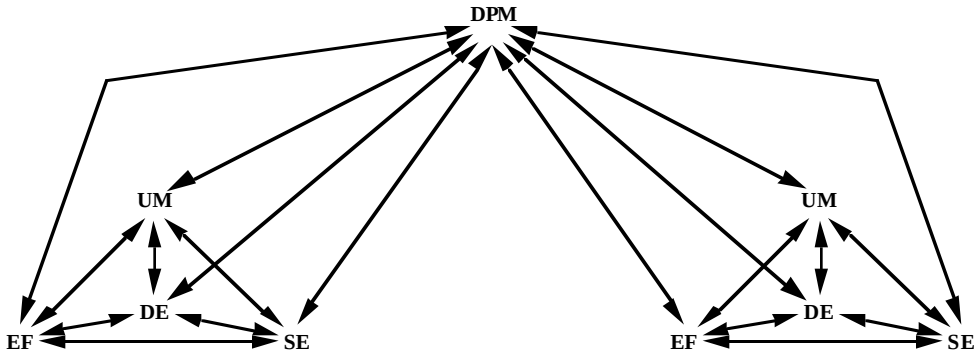

Figure 6 Communication during project creation

6.2 Project creation scenario

The component OPC of the design project manager (DPM) has the goal to design the interior of an aircraft (1). To reach this goal, DPM needs help. Thus, his component GP (part of CM) is activated to generate the project. Immediately, PPC (part of GP) is activated to determine which activities are needed to reach the goal and which possible team members for the project (2) can be found. For this purpose DPM requests possible participation from design engineers, electrical engineers, systems engineers, unit managers, styling specialists and tool experts.

time point	agent	component	sub-component	subsub-component	links
1.	DPM	OPC			self_info_for_cooperation
2.	DPM	CM	GP	PPC	required_project and recipe_to_be_prepared
3.	DPM	AIM			needed_info_on_other_agents, required_info_on_other_agents, info_to_be_communicated, communication_to_be_performed, and the links between DPM and the other agents
4.	other	AIM			incoming_communicated_info
5.	other	OPC			communicated_info_to_OPC
6.	other	AIM			link from OPC to AIM, communication_to_be_performed, and the link between the agent and DPM
7.	DPM	AIM			incoming_communicated_info, communicated_info_to_CM
8.	DPM	CM	GP	PPC	communicated_info_to_CM, info_on_other_agents, and info_on_other_agents_to_PPC
9.	DPM	CM	PG	GMR	prepared_project
10.	DPM	AIM			info_for_agents, required_info_on_other_agents, info_to_be_communicated, communication_to_be_performed, and the links between DPM and the other agents
11.	other	AIM			incoming_communicated_info
12.	other	OPC			communicated_info_to_OPC, the link from OPC to AIM, communication_to_be_performed, and the link between the agent and DPM
14.	DPM	AIM			incoming_communicated_info and communicated_info_to_CM
15.	DPM	CM	GP	GMR	info_on_other_agents and info_on_participants.
16.	DPM	AIM CM	MP		info_for_agents, commitments_to_output, info_to_be_communicated, info_for_agents, and own_generated_project

Table 7 System trace: project creation

The requests are handled by DPM's AIM component (3). Each of these agents receives the request through its own AIM component (4), and considers the request for possible participation in its own component OPC (5). Each agent's AIM component returns an answer to the request (6). DPM receives the agents' responses (in his AIM component) (7). The replies are forwarded to the PPC component, which continues the preparation of project commitments in interaction with the possible participants (iterating steps 3 through 8). The information on the project activities and the willing participants is sent to GMR (part of GP). This component is responsible for the creation of the final recipe. This task involves frequent contact with the willing participants. Again this contact is handled by the AIM components of the agents (10,11). The OPCs of the willing participants check to see if the activities assigned to them fit in their own schedules (12). Information on the success or failure of their scheduling is sent by their AIM component (13) to the AIM component of DPM (14), which forwards it to GMR (15). By iterating steps 10 through 15, GMR creates a final recipe.

The resulting recipe includes the global goal (i.e., aircraft to be designed given global requirements and specifications) and recipe elements. A recipe element related to the design of a unit includes the following information:

- the specific requirements and specifications for the unit to be designed (based on the initial design of the whole aircraft),
- one unit manager (UM),
- one design engineer (DE),
- one electrical engineer (EE), and
- one systems engineer (SE).

The resulting recipe is sent to each of the unit managers by AIM (16). The CM component of DPM makes sure that the resulting recipe will be monitored by its subcomponent MP (16).

After the unit groups have been formed the unit managers schedule the design process of their unit, following a similar pattern.

7 Discussion

Multi-agent literature focusses on modelling interaction between agents, most frequently based on informal models of interaction; see (Wooldridge and Jennings, 1995). The formalization of Jenning's (1995) proposal in the DESIRE framework explicitly contains a task decomposition, a specification of the information exchange and task sequencing, details lacking in Jennings' original model.

In the process of formalizing Jennings' model the following assumptions, underlying the approach, were found:

- communication is assumed to be fool proof.
- message delay time is assumed to be known to all agents.
- agents are assumed to have mutual beliefs: one level of nesting deep (everyone knows that everyone knows).
- a global clock is assumed.
- agents are assumed to be able to predict, with a reasonable degree of accuracy, the time taken to execute each of their domain level activities.
- durations of actions are assumed to be fixed and known to all.

Some gaps in Jennings' model have been discovered and solutions were chosen for the formal model. For example with respect to the generation of recipes: the original model only specifed what to send to possible participants, not when. If the sequencing is unfortunate, the process might not terminate and previous commitments might have to be given up in favour of new ones. For the component generate_project in the formal model a specific technique has been specified that can be used for the generation of recipes. With this technique it is clear when and what to send to possible participants. The technique can be proven to terminate. Of course, this approach is but one solution for that problem. Other negotiation strategies can also be formalized in the generate_project component.

Furthermore, in the original approach the priorities of actions were static and predetermined. In the formal model, actions can have priorities that are determined and changed in a dynamic manner. The effect of a change in priorities is discovered by the component monitor_project of one of the participating agents and, if necessary, the component generate_project of that agent reconsiders the project.

In the original model only total success and total failure could lead to the reconsideration of a recipe. In the formal model it is possible that a recipe is reconsidered on the basis of partial success or partial failure, leading to different actions and new goals.

The generic specifications of the model can be used in diverse project coordination situations, instantiated for the specific domain of application. This paper contains an illustration of the application of the model for cooperation in aircraft interior design.

To conclude, by formally specifying not only the knowledge involved, but also the task sequencing, and the types of interaction and coordination patterns required during project coordination, Jennings' model has been refined and generalized. Furthermore, more detailed insight has been acquired in the required type of support (for example, types of verification and validation).

Acknowledgements

Discussions on the cooperation model with Nick Jennings were very useful to the authors.

References

Bahler, D., C. Dupont, J. Bowen (1994). An axiomatic approach that supports negotiated resolution of design conflicts in concurrent engineering. In: J.S. Gero, F. Sudweeks (eds.), *Artificial Intelligence in Design*, Kluwer Academic Publishers, Dordrecht, pp. 363-379.

Brazier, F.M.T., B. Dunin-Keplicz, N.R. Jennings, J. Treur (1995). Formal Specification of Multi-Agent Systems: a Real World Case. In: V. Lesser (ed.), *Proc. of the First International Conference on Multi-Agent Systems, ICMAS-95*, MIT Press, pp. 25-32.

Brazier, F.M.T., B. Dunin-Keplicz, N.R. Jennings, J. Treur (1996). DESIRE: modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Information Systems*, M. Huhns, M. Singh, (eds.), special issue on Formal Methods in Cooperative Information Systems, 1996, to appear.

Brazier, F.M.T., C.M. Jonker & J. Treur (1996). Formal specification of a model for cooperation based on joint intentions. Technical Report, Vrije Universiteit Amsterdam, Department of Mathematics and Computer Science

Brazier, F.M.T., J. Treur, N.J.E. Wijngaards and M. Willems (1995). Formal specification of hierarchically (de)composed tasks. In: B.R. Gaines and M.A. Musen (eds). *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW '95*, 1995, Volume 2, pp. 25/1-25/20. Calgary: SRDG Publications, Department of Computer Science, University of Calgary.

Brazier, F.M.T., J. Treur, N.J.E. Wijngaards and M. Willems (1996). Temporal semantics of complex reasoning tasks. In: R. Albrecht, H. Herre (eds). *Proc. of the Int. Workshop on New Trends in Theoretical Computer Science*, Oldenburg Verlag, Munich-Vienna.

Cutkosky, M., R. Engelmore, R. Fikes, T. Gruber, M. Genesereth, W. Mark, J. Tenenbaum, J. Weber (1993). PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer 26, Special Issue on Computer Support for Concurrent Engineering*, pp. 28-37.

Dunskus, B.V., D.L. Grecu, D.C. Brown, I. Berker (1995). Using single function agents to investigate conflict. *AIEDAM 9*. Cambridge University Press, pp. 299-312

Jennings, N.R. (1995). Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, *Artificial Intelligence Journal* 74 (2)

Jennings, N.R., J. Corera, I. Laresgoiti, E.H. Mamdani, F. Perriolat, P. Skarek, L.Z. Varga (1995). Using ARCHON to develop real-world DAI applications for electricity transportation management and particle accelerator control, *IEEE Expert - Special Issue on Real World Applications of DAI.*

Klein, M (1995). Conflict management as part of an integrated exception handling approach. *AIEDAM 9*, Cambridge University Press, pp. 259-267.

Langevelde, I.A. van, A.W. Philipsen and J. Treur (1992). Formal specification of compositional architectures, in B. Neumann (ed.), *Proceedings of the 10th European Conference on Artificial Intelligence, ECAI'92* , John Wiley & Sons, Chichester, pp. 272-276.

Petrie, C. (1994). Design space navigation as a collaborative aid. In: Gero, J.S. (ed.), *Artificial Intelligence in Design '94, Proceedings AID '94.* Kluwer Academic Publishers, Dordrecht, pp. 611-623.

Wooldridge, M., N.R. Jennings (eds.) (1995), *Intelligent Agents*, Proc. of the ECAI'94 Workshop on Agent Theories, Architectures and Languages, Lecture Notes in AI, vol. 890, Springer Verlag.